Deep Learning-based Transmitter identification on the physical layer

Cyrille Morin, Leonardo Cardoso, Jakob Hoydis, Jean-Marie Gorce

Deep Learning-based Transmitter identification on the physical layer

Cyrille Morin, Leonardo S. Cardoso, Jakob Hoydis, *Senior Member, IEEE,* and Jean-Marie Gorce

***Abstract*—An essential part of most wireless communications systems is the identification of a transmitter by a receiver. Being able to identify a transmitter at the physical layer gives context to the communication itself, but is also an important building block for more advanced techniques such as physical layer security. It can also be used to reduce overhead in the transmission of small packets. Previous works have shown the usefulness of applying deep learning to this task, however, as seen in those works, channels characteristics tend to capture the fingerprinting abilities of deep learning systems, rendering them very sensitive to changes in the radio propagation environment. This work focuses on reducing the impact of channel effects on identification performance and generalisation to changing conditions, something that has been little addressed in the literature, and show that increasing channel variations in the data used to train a neural network can increase its resiliency to channel modifications, leading to a gain of up to 21.3 percentage points in accuracy compared to the naive approach found in the literature. The datasets collected for this paper are available online, as well as the tools to collect new ones, in the hope that they can be reused by the community.**

***Index Terms*—Fingerprint identification, Learning systems, Radio transmitters, Data acquisition, Test facilities**

I. Introduction

Transmitter identification has been a crucial topic since the dawn of radio communications. Unlike wired communications, where communication integrity is ensured by the physical medium, the broadcast nature of electromagnetic waves requires to securely identify the transmitter of a signal. Since World War II, radio identification is mainly based on cooperation with the transmitter, thanks to an identification code. This method of identifying transmitters is inherently prone to problems since it depends on decoding a portion of the signal itself and also on *trusting* that the actual source of the signal is not trying to impersonate a trusted transmitter.

The problem of identifying a transmitter has become even more important nowadays due to

two factors. First, the widespread availability of low cost, programmable radios, such as software defined radios (SDR), has opened the radio spectrum to a large public. While still not very widespread, security attacks such as man-in-the-middle (MITM) are utterly possible with SDR technology [1], or simply by altering medium access control (MAC) layers of popular radio devices [2]. Second, the sheer number of radio devices accessing the spectrum is enormous and tends to become even greater with the undergoing massive deployment of internet of things (IoT) devices. To identify all these devices with unique identification codes means that very long codes must be used. Such codes will take up valuable resources and, as already stated, are prone to attacks by malicious transmitters, see for instance [3].

To solve these problems, other means of identifying transmitters must be sought. Preferably, these identification techniques should not depend solely on transmitted identification codes but rather rely on physical characteristics of the transmitting hardware. This is indeed possible since transmitters are built with discrete components which are produced to a certain tolerance around their nominal values, creating a slight but perceivable variation of the radio signal, even between devices of the same brand and model. The compound effect of these and other factors can be seen as a fingerprint of the transmitter, that could be effectively recognised by a receiver.

There are two main strategies available to perform an identification task, relative to the way known signals are memorised to be identified later:

- Comparison: The identifying system has 2 inputs: the signal to identify, and a known one. It then outputs a score, rating how close the two signals are to each other. This is repeated with several known signals to find the closest match. In that case, users are recalled by their stored reference signals, not by the system itself. If characteristics tend to change over time, the reference signals can be updated regularly to keep up with the changes. This approach makes it simpler to work with new users, simply by storing new reference signals, that might require large amounts of memory, scaling with the number of transmitters to be identified.
- Classification: The identifying system has only one input: the signal to identify. It then outputs a score rating how much each possible candidate matches to the received signal. In this case, users are recalled by the identification system itself (e.g. as weights in a NN), so updates are harder to accomplish to accommodate for variations of characteristics or new users. On the other hand, identification is done in one step instead of computing a score for ev-

ery reference signal, hence shortening the processing time and amount of memory required.

The literature on physical layer transmitter identification covers both of these strategies, as can be seen in Table I. On the comparison side, most approaches directly use the channel effects for identification [5]–[10], while [11] does not include them in their simulation. Most interesting in the scope of the present paper, the authors of [12] introduce a new feature: Amplitude of Quotient (AoQ) that looks at the variation of a Wifi preamble signal over two consecutive frames. This allows to greatly reduce the channel dependency since, from one channel to the other, the variation between two concurrent frames stays the same. Unfortunately, this approach is difficult to implement in an Internet of Things (IoT) scenario where devices can transmit isolated packets with potentially big time gaps in between them. For this case, a one-shot identification system is needed.

Most of the works on the subject focus rather on classifiers, with many experimental setups, over a wide range of hardware platforms (phones, USRPs, Wifi, IoT and Zigbee devices), using a variety of preprocessing techniques and mostly using the steady state features of transmitted packets. Although the range of demonstrated applications seems very promising for real-life usage cases, the vast majority of studies do not consider the effects of a non-static channel on the identification performance. The effects of channel variations can be first seen in [31], where, even though the authors show the stability of learned features (involving 54 transmitters) by testing their network with new data 18 month after it was trained, they encounter a drop in performance when changing the location of the transmitters, due to a drastic change of the channel.

The authors of [32], [33] created a dataset using 16 USRP devices in a realistic room used to train a convolutional neural network (CNN) with good results. Unfortunately, any change in the environment (position of chairs or people moving) prevents the generalisation of the neural network to a new dataset. To counter this effect, additional artificial impairments are embedded in the transmitted signals to improve performance and a protocol is devised to choose those impairments, in a similar manner as for cryptographic keys. To the best of our knowledge, this paper is the only one that provides the dataset they use in their experiments. To the same end, the authors in [34] introduce a FIR filter in each transmitter and use a feedback link from the receiver to optimise their parameters to increase identification accuracy at the receiving CNN. The latter two approaches effectively increase robustness to channel changes, but they reduce the strength of their security claims: as with the

TABLE I

COMPARISON OF LITERATURE PAPERS

Paper	Approach	Channel effects	Main feature	Preprocessing	Devices	Algorithm	Type	Dataset
[4]	Analytical	No	Transient	No	28	N/A	VHF	Exp
[5]–[9]	Comparison	Used	CSI	N/A	2	Expert	N/A	Simu
[10]				Chan features	?	GMM	USRP	Exp
[11]		No	IQ imbalance	No	5	CNN	N/A	Simu
[12]		Yes	Steady state	AoQ	15	DNN	Wifi	Exp
[13]	Classification	No	Transient	Hilbert	8	PNN	Wifi	Exp
[14]				Bispectrum	3	SVM	USRP	Both
[15]			Many	Feat. extraction	4	NN,SVM		Exp
[16]			Amplifier	FFT	7	NN		Both
[17]			Steady State	Hilbert	5	CNN	N/A	Simu
[18]				Feat. extraction	10000	DNN	N/A	
[19]				VMD	20	LSVM	Phones	Exp
[20]				Handcrafted	6	Expert	Custom	
[21]				STFT	4	SVM	USRP	
[22]				Minkowski	9	SVM,k-NN		
[23]				Recurrence plot	11	CNN		
[24]				No	5	ML	USRP	
[25]					6	NN	Zigbee	
[26]					6	ML	USRP	
[27]					8	k-NN		
[28]				Error signal	7	CNN	Zigbee	
[29]				Denoiser	27			
[30]				Multisampling	52			
[31]		Yes[1]		CTF	54	NN		
[32], [33]		Yes		No	16	CNN	USRP	
[34]					20			
[Ours]					21			

current approach, a malicious user could tune its own system as easily as a registered one.

Two papers that stand out from Table I are [4] which is a study of possible characteristic features of transient signal in VHF radio, and [35], where the authors, instead of focusing on getting the best possible accuracy on their dataset, attempt to predict the size needed for said dataset. The precise numbers reached in that paper are specific to their neural network

design and scenario but they provide a rule-of-thumb for choosing dataset sizes.

The approach proposed in this paper is to train a CNN classifier using raw IQ samples without preprocessing and from experimental datasets to ensure that all possible features in the signal can be used by the neural network. The network is made to cope with the channel variations effects natively, without having to introduce any form of additional information at the transmitter, by purposely introducing channel variations in the training dataset.

The remainder of this paper is divided as follows: Section II specifies the problem herein addressed, then Section III describes the implementation chosen to tackle it, from the experimental dataset gathering, to the neural network (NN) architecture used for identification. Section IV shows the results obtained using this implementation, and Section V analyses some specific effects of synchronisation between transmitter and receiver. Finally, Section VI concludes this paper and elaborates on possible future contributions.

II. Problem Formulation and Classifier Selection

A. Problem statement

The transmitter identification problem consists of a set of N transmitter *nodes*[2], and a set of M *identification receivers*[3]. The k^{th} node sends a complex discrete sequence, $x_k(t)$ and the i^{th} identification receiver receives a complex discrete sequence $y_i(t)$. Throughout this work, we refer to "MonoRX" in the case where $M = 1$ and "MultiRX" in the case where $M = N$. In regular systems, a part of $x_k(t)$, namely the identification field in the packet header, is used to identify the node. Unlike those systems, here we consider that $x_k(t)$ does not contain any information enabling the identification of the node, so x can be considered independent of k and we define $x(t) = x_k(t)\ \forall k$.

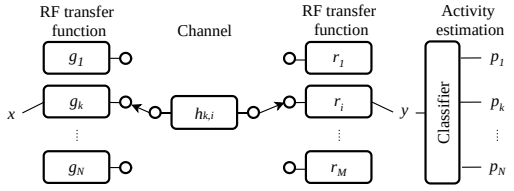

Fig. 1. System model

The sequence x is then processed by the radio frequency (RF) chain of the k^{th} radio device, whose overall RF transfer function g_k is unknown and can be non-linear. This RF transfer function corresponds to the compound effect of all signal formatting such as conversion from digital to analog domain, mixing, filtering and amplification.

The produced signal is transmitted through a channel $h_{k,i}$ between node k and the identifica-

[2]Henceforth all transmitters are called *nodes*.

[3]Henceforth the receivers are called *identification receiver*.

tion receiver i, all channels having unknown input responses. The identification receiver then applies an RF transfer function r_i which, as for the node, can be non-linear. This leads to the observation of a sampled baseband signal $y(t)$, from which a decision the Id of the active node has to be taken. The overall model can be seen in Fig. 1. Note that in this setup, possible collisions are not considered and during each slot only one node and one identification receiver are active, chosen randomly, leading to the following system equation:

$$y(t) = r_i(h_{k,i} * g_k(x(t))) \tag{1}$$

With k from $K \sim \mathcal{U}_{\mathbb{N}}(0, N)$ the active node and i from $I \sim \mathcal{U}_{\mathbb{N}}(0, M)$ the active receiver.

Now the question addressed in this paper can be stated: Is there enough information in the observed signal $y(t)$ to blindly find k without using the channel signature, but exploiting only the signature of the RF transfer functions g_k ?

The objective is then to design a system able to compute classification functions $\tilde{p}_k = f_k(y(t))$ where $\tilde{p}_k$ the estimated probability that the k^{th} node is active. To evaluate the performance of such a multi-class classification system, it is standard practice to use the categorical crossentropy loss function [36]:

$$\mathcal{L} = -\sum_{a=1}^{N} \mathbf{1}_{[a=k]} \cdot \log\left(\tilde{p}_k\right). \tag{2}$$

B. DL based classifier

A system able to minimise (2) can answer to the first half of the question at hand: Is there enough information in the observed signal y to blindly learn about the ID of the active node? The other half comes from a careful choice of dataset, as described later.

To do that, the approach chosen is to train a NN classifier operating on the raw observed signal y and directly outputting estimated activity probabilities $\tilde{p}_k$ for all possible nodes. The use of deep learning is fully justified: first the features of the function g_k we want to rely on are unknown and expected to be non linear, and an experimental setup using the Future Internet of Things / Cognitive Radio Testbed [37] (FIT/CorteXlab), as described in the next section, allows to get a large database of real signals for training, which is the first of its kind at the best of our knowledge.

No preprocessing is done on the raw data to avoid filtering any unexpected, but useful feature that may be present in the dataset.

The selection of a network architecture type relates to the assumptions made on the characteristics of the input signal. In this case, with the same reasoning as with preprocessing, the only assumption is that the signal is correlated in time, leading to the selection of a 1D CNN type network. This is because small scale memory effects can be expected, e.g. in the

amplification stages. On the other hand, that assumption reduced by having the last layers of the network be general purpose dense neural network (DNN).

The network is trained using the loss function (2), but the use of that loss function alone does not allow to discriminate between the effects of g_k and $h_{k,i}$ since they both depend on k. To permit such a discrimination, a training dataset will be specifically built in the next section to reduce the impact of $h_{k,i}$ in the learning process.

III. Implementation

To properly train a neural network for the identification task, a large dataset that is properly unbiased and correctly labelled is required. In this section, the data collection methodology used to achieve a suitable sample set for our identification needs is described: first, the experimentation room is presented, then the overall system architecture controlling the operation of the nodes and receivers. This is followed by the description of the scenarios for gathering data, allowing to study the impact of different parameters on the identification performance. Finally, the exact NN architecture and training on the obtained datasets is presented.

A. Experimentation setup

1) Experimentation room: The complete system, comprised of a full set of nodes and the identification receiver, is tested within the FIT/CorteXlab testbed deployed in an isolated and semi-anechoic experimentation room (which shields the experiment from any external interference), of about 180 m^2. Using FIT/CorteXlab allows to control the propagation environment as well as the interference profile, which in turn enables full control of the generated datasets. The nodes, as well as the identification receiver, are implemented on a software defined radio (SDR) of the NI USRP-2932 type. An optional synchronisation can be achieved over all SDRs, effectively synchronising all sampling clocks of all SDRs to evaluate the impact of synchronisation errors as detailed in Section V. Since the propagation environment of the FIT/CorteXlab experimentation room is rather static, a metallic covered Turtlebot robot, as seen in Figure 2, can be activated and moves according to a random-walk model. This allows to create time varying alternative propagation paths leading to channel diversity inside the room.

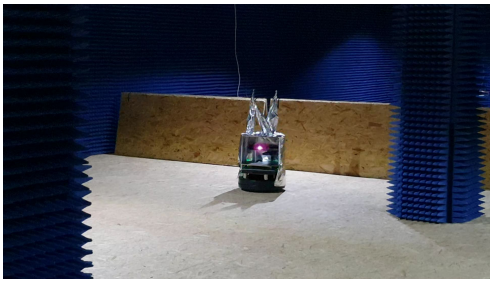

Fig. 2. The Turtlebot robot with the metallic sheets, inside of FIT/CorteXlab

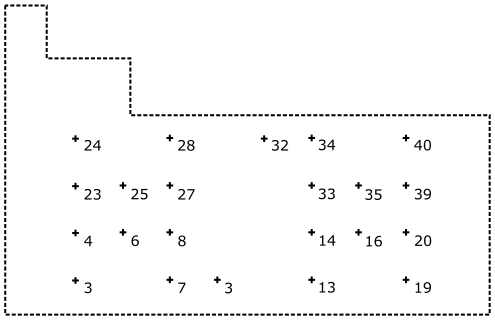

Fig. 3. FIT/CorteXlab experimentation room plan and node locations

The nodes are distributed as in Figure 3, where the position of all USRP nodes' antennas are indicated by the crosses with the node numbers, and the footprint of the FIT/CorteXlab experimentation room is delimited by the dotted line. The nodes' antennas are distributed onto a grid with a step size of 1.8 m^2. Among all active USRPs in the experimentation room, one takes the role of the identification receiver while the remaining behave as nodes. The choice of which node takes which role can be defined in the experimentation scenario description file.

Previous works characterised the channels in FIT/CorteXlab, such as [38] and [39]. The work in [38] has shown that the channels inside the FIT/CorteXlab experimentation room are indeed static with respect to the fading scenario as a whole (path-loss, shadowing and small-scale fading). This result is expected since all nodes have fixed positions and there is no interference from outside environmental changes. In [39], the authors provide a measurement of the power-delay profiles observed in the room. Measurements made at a sampling frequency of 10 MHz (low resolution measurements for the dimensions of the room) showed that the channels are essentially flat fading, with very little delay spread. However, even if the FIT/CorteXlab experimentation room is partially covered with electromagnetic (EM) absorbing foam (roof and walls), reflections off the floor and metallic structures occur, creating multipath but with limited delay.

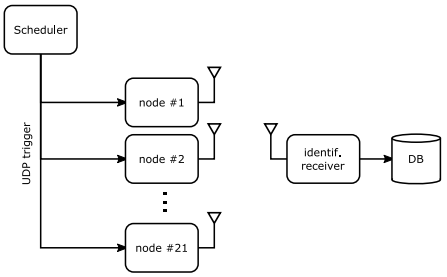

Fig. 4. Overall data collection topology (MonoRX)

2) System architecture: In each data collection experiment, 22 or 23 devices are involved: one *scheduler*, one *identification receiver* and 20 or 21 *nodes*, as depicted in Figure 4.

The scheduler, as the name implies, orchestrates the transmission of packets over all transmitting nodes, guaranteeing an interference free scenario. It sends a trigger signal to a specific node that initiates a packet transmission. While the scheduler is assigned to a SDR node, it does not need a radio transmitter. Wired Ethernet

connections are used to trigger the transmitting nodes. The scheduler sends a trigger signal every millisecond to a randomly chosen node via UDP, which ensures that transmissions are made temporally close to each other. This is essential for dynamic channels settings. As such, packets from a single node are spread over the duration of the experiment, and one specific realisation of the channel sees transmissions from more than one node.

The nodes' USRPs are set to "burst mode", to largely reduce the possibility of the oscillator leakage noticed in [16]. This means that the amplifier of an USRP is turned off when not transmitting. Consequently, the radio frequency (RF) circuitry needs time to wake up and stabilise before transmission. Hence, a frame is prepended with 3000 zeros, corresponding to a delay of 0.6 ms.

On the identification receiver side, the USRP remains in listening mode for the duration of the experiment. In order to properly separate the packets from noise, and properly label the received packets' transmitter a robust detection mechanism is required. This detection mechanism is based on a time synchronisation scheme coupled with an identification header implemented as described next.

At the nodes, the packets to be transmitted are encapsulated into a *carrier frame* with the following elements:

- A Zadoff-Chu sequence preamble for frame detection and time synchronisation;
- An orthogonal frequency-division multiplexing (OFDM) frame header created using the standard GNU Radio OFDM blocks and containing the node index for data labelling;
- A user-defined payload made from a known quadrature phase shift keying (QPSK) modulated sequence, random modulated bits or uniform noise, as detailed in Subsection III-B1. This part of the overall frame never contains transmitter specific information.

A guard sequence longer than the delay spread of the channel is added between these parts to prevent interference. The overall transmitter GNU Radio scheme is presented in Figure 5.

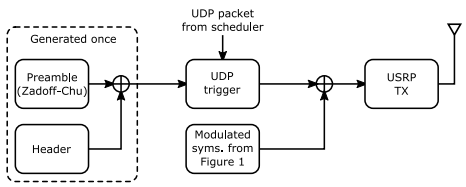

Fig. 5. Simplified transmitter flowgraph

At the receiver side, frame detection and time synchronisation are done with a correlator that sweeps the signal looking for the Zadoff-Chu sequence. The next 1040 samples, corresponding to the header, payload, and guard intervals are then forwarded to the header and payload extraction blocks. A standard OFDM receiver then decodes the header and forwards

it to a block that uses the transmitter index to send the payload to the corresponding file. The overall receiver chain is presented in Figure 6. Finally, the recorded signal, henceforth denoted *example*, is 600 complex samples long, larger than the payload, whose size is 560 complex samples. This oversized cut is exploited to record a small amount of background noise before and after the payload, to ensure recording the start and end of the payload. A standard size dataset is comprised of about 50000 recorded examples for each of the 21 nodes. The authors in [35] suggest, as a rule-of-thumb, to have a number of examples at least 10000 to 30000 times the number of transmitters. So a standard size dataset is comprised of about 50000 recorded examples for each of the 21 nodes.

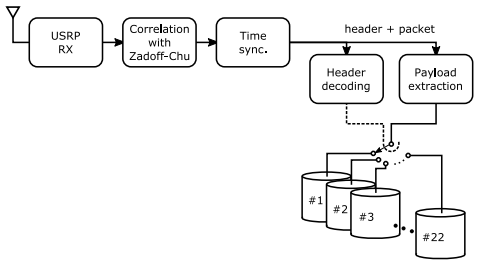

Fig. 6. Reception flowgraph.

B. Dataset scenarios

Each experiment is characterised by a tuple of parameters: the payload type, the transmission setup and the number of identification receivers. Such a tuple is referred as a *Scenario* in the rest of this paper.

1) Payload: Three types of payload ($x(t)$ in the system model) are studied:

- *Static*: A predefined sequence of bits modulated in QPSK as defined as the 802.15.4 protocol preamble. A fixed sequence reduces the variability in the examples the CNN trains on, thus reducing the difficulty of its task, while remaining a realistic setup: in such a case, the frame preamble would be used, as it still needs to be transmitted.
- *Random*: Bits from a random source modulated in QPSK. This represents the case where the identification is made using the frame section that contains user data.
- *Noise*: A complex random uniform sample sequence as a worst case scenario. In this case, no pattern coming from a modulation scheme can be used as reference to help measure impairments. It is used as a lower bound on what is achievable by the network architecture regardless of the actual type of modulation used.

2) Transmission setup: As explained previously, the propagation channel inside FIT/CorteXlab is static and the various devices are fixed on the ceiling and distant from each other, so the channel is quite different from one node to the next. Yet the goal is to be able to identify nodes based on their hardware characteristics and not the channel effects. To

remedy this, 3 transmission setups are defined:

- *Plain*: The basic case where nothing is done to mitigate the channel biases. Everything is static, the experimentation room and transmission parameters, thus the amount of variability in the system is reduced and the identification task is made easier. This mode serves mostly as a benchmark to compare to the other modes and measures the tradeoff between scenario complexity and learning ability.
- *Varying*: In this case, the amplitude of the payloads to be transmitted by the nodes is scaled by a factor that changes over time before emission by the USRP. This method is preferred over changing gain values for the amplifier in the USRP because it eliminates the need to wait for amplifier stabilisation. It allows the emulation of path loss variations for every node without having to physically move anything in the experimentation room.
- *Robot*: The robot described previously is introduced and set to randomly move around. This mode also include amplitude variation and is the most complex and realistic scenario but is also the most time demanding to run, and so is only used with the static payload.

3) Number of receivers: In order to increase even more the channel variations and to reduce the possibility for the receiver to learn from the channel properties, the *MultiRx* setup is proposed where we merge the signals observed from several devices acting alternatively as identification receivers. This *MultiRx* setup, in contrast to the *MonoRx* setup, allows the collection of payloads from 20 different receivers for each transmitting node. It is done by collecting the equivalent of a set of 21 MonoRx experiments, each with a different single receiver. That means that one specific payload is recorded only by one receiver, and not the 20 possible. The neural network defined in the next subsection always uses only one payload, from one receiver whose Id is not given, and it does not know what mode was used.

Practically speaking, another difference between the two modes is the fact that the node number 3 is not used in MultiRx. FIT/CorteXlab contains 22 USRPs devices, so, in MonoRx, 21 act as transmitting nodes while one preselected USRP acts as the receiver. But in MultiRx, the roles are permuted and so the 22 devices act alternatively as transmitters or receiver. However, to facilitate the comparisons between the two modes, the same number of transmitters is used, namely 21.

The datasets collected and used in this paper

[3]In this case, channel effects are observed but not corrected or accounted for.

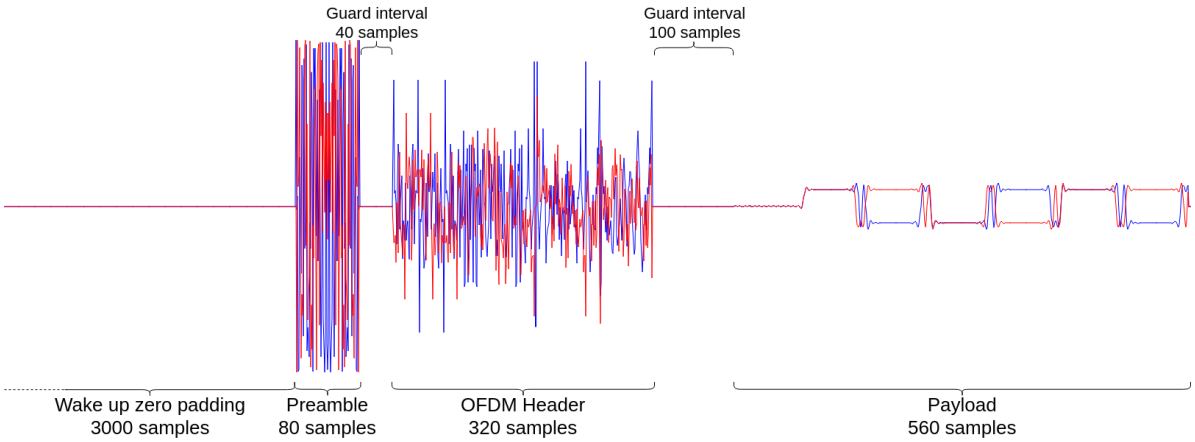

Fig. 7. Frame samples sent to USRP for emission, with zeroes for amplifier wake up, preamble, header, and payload with guard intervals.

TABLE II

CONSIDERED PAYLOAD CHARACTERISTICS

Parameter	**Value**
N	21
M	1 (*MonoRX*), 21 (*MultiRX*)
Frequency band	ISM 433 MHz
Signal bandwidth	2.5 MHz
Waveforms	QPSK (RRC filtering), Uniform
Samples/symbols (QPSK)	2
Total samples	560

are available online [40] in the hope that they can be reused by the community.

C. Learning architecture

1) Network architecture: As shown in Fig. 8, the neural network used is based on a CNN architecture: five 2D convolution layers with a max pooling layer between each of them, a flattening step and six fully connected layers before the final softmax output.

TABLE III

DETAILED NETWORK LAYERS DESCRIPTION

Layer	Size	Kernel	Activation
Conv_0	8	[6, 2]	Elu
MaxPool_0		[2, 1]	
Conv_1	16	[4, 1]	Elu
MaxPool_1		[2, 1]	
Conv_2	32	[4, 1]	Elu
MaxPool_2		[2, 1]	
Conv_3	64	[4, 1]	Elu
MaxPool_3		[2, 1]	
Conv_4	128	[4, 1]	Elu
MaxPool_4		[2, 1]	
Flatten	1920		
Dense_0	512		Elu
Dense_1	256		Elu
Dense_2	128		Elu
Dense_3	64		Elu
Dense_4	32		Elu
Dense_5	16		Elu
Output	21		Softmax

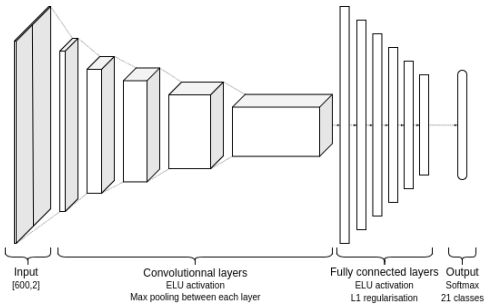

Fig. 8. Neural network architecture

The chosen method for handling the complex-valued input samples is to represent them by their Cartesian coordinates, treating the real and imaginary parts as independent input dimensions. The vector of 600 complex values becomes a matrix of 600×2 real numbers used as a 2D image by the convolution layers. The network outputs a probability vector over the 21 possible transmitters.

2) Training phase: Each dataset is randomly shuffled before training and split in a standard 70/10/20 distribution for training, validation and testing. These will be called *Training slice*, *Validation slice*, and *Test slice* in this paper instead of the usual term *set* to reduce confusion with dataset and set of datasets. Each scenario is used to train a different instance of the architecture presented above. Training examples are presented in mini-batches of 128 examples over 30 epochs for the MonoRx scenarios and 100 epochs for the MultiRx ones. The Adam optimiser is used with a learning rate of 0.001, tasked with minimising a categorical crossentropy loss between labels and predictions.

Hyperparameter tuning was done using a hard to learn dataset: MultiRx setup with amplitude variation and random payload. for this, a ten epoch long training was done for various parameters (learning rate, batch-sizes, layers) and the best performing hyperparameter set was retained.

IV. Results

A. Payload type

In this section, the impact of the payload type on the classification performance is assessed experimentally. We perform a series of test corresponding to different scenarii as described in Section III. The three kind of payloads (static, random, noise) are tested, combined with the two channel conditions (plain or variable) and with the MonoRx or MultiRx configuration. The results are presented in Fig. 9. These graphs provide the identification accuracy over the test slice of the training dataset for each combination (payload,channel,receiver). Clearly the learning is more accurate when the payload is static (left figure). This result indicates that such identification task on ambient signals,would be more efficient if performed on a fix preamble or header, rather than on the data payload.

When the channel variability is increased by using varying channel gains (var) and MultiRx, a significant accuracy loss is observed (last

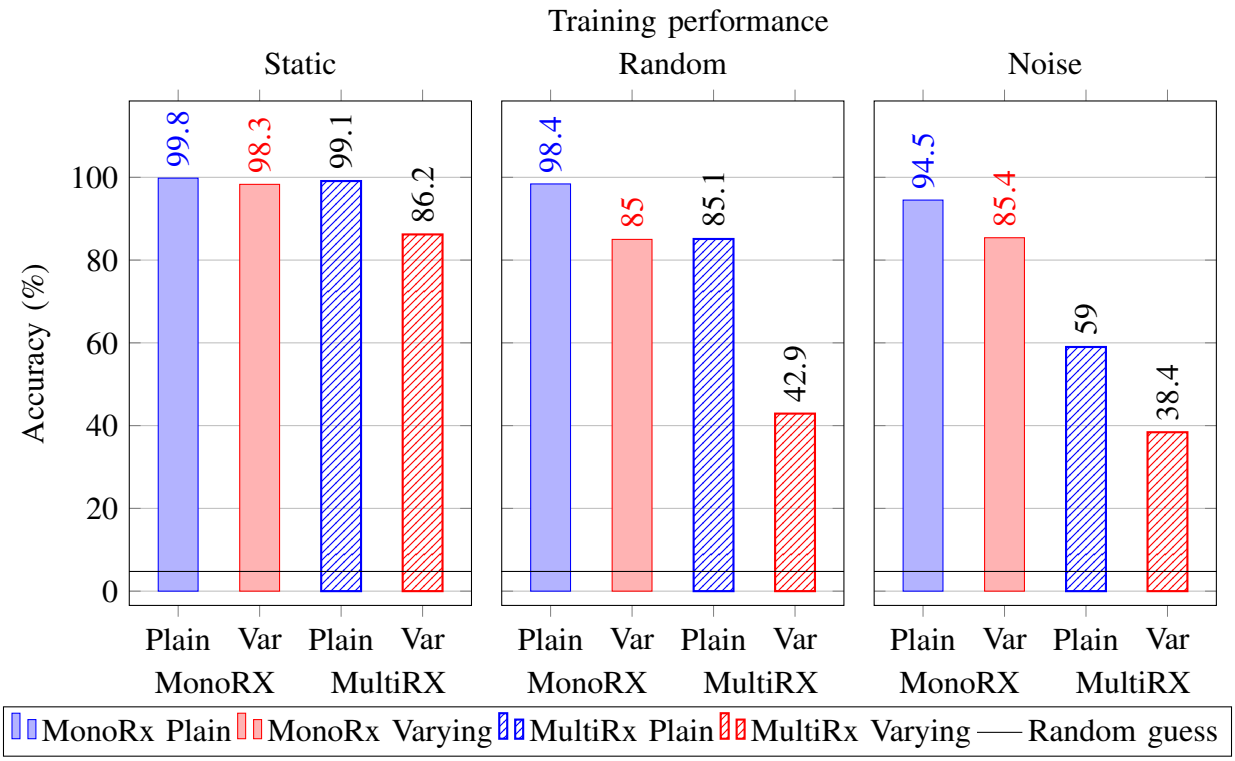

Fig. 9. Accuracy reached by networks trained on plain or varying amplitude scenarios and the 3 signal types. Accuracy is measured on the test set from the training dataset.

column in left figure), but the accuracy remains high at 86%.

With the two other payloads (middle and right figures), the learning capability reduces drastically when channel varying and MultiRx setups are used (last column, with 38%). Clearly, the high accuracy observed with plain and MonoRx setup with these payloads may be achieve thanks to the channel signature rather than the radio node itself, explaining why when MultiRx and variable channel are used, the accuracy collapses (it still remains an order of magnitude above random guess). It is worth mentioning that the scenarios with the Random and Noise payload need a dataset twice as large to avoid overfitting on the training slice. This behaviour is not surprising since these scenarios combine the most variability in payload, channel and also receiver impairments.

For all these reasons, in the rest of this work, a Static payload will be used.

B. Impact of the transmission setup

In this section we assess the impact of the transmission setup on the identification capability.

According to the results of the former section, the payload is static in these experiments.

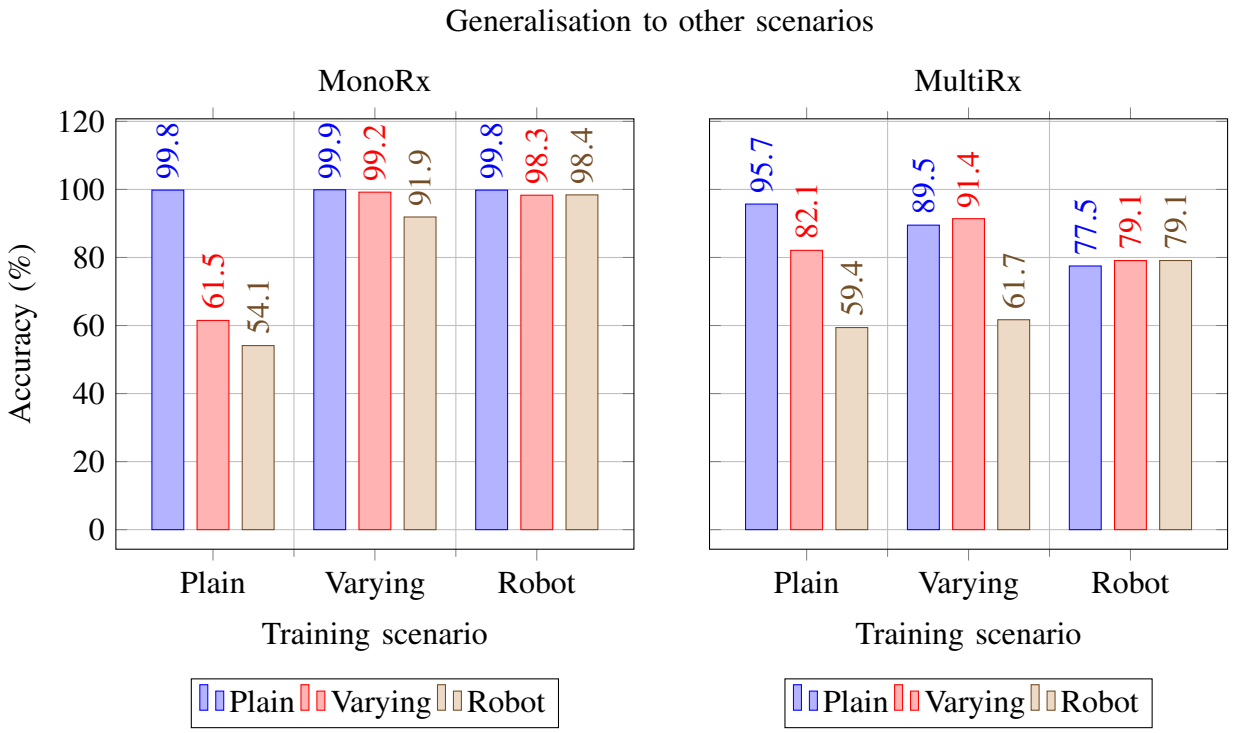

Fig. 10. Accuracy of the networks trained on three different scenarios (x axis legend) and tested on other datasets of the three kinds (resp Plain, Varying, Robot) as indicated by the bars' colors. In these experiments, the payload of all packets was static and the environment in the shielded room remained unchanged.

The identification performance over three scenarios, corresponding to three channel conditions referred as Plain, Varying or Robot, are evaluated. Note that in the scenario labelled Robot, the varying channel conditions are used. Three learning dataset are thus built, one for each of the three scenarios, and used to train three networks. Then, the three networks are used independently on new datasets and the identification accuracy is presented in Figure 10.

The left Figure presents the results for the MonoRx scenario. The blue bars correspond to the accuracy obtained on the test set Plain, with the networks learned respectively on the three learning sets: Plain, Varying and Robot. The identification accuracy is high in all cases. On the opposite, the network trained in Plain conditions is not able to efficiently identify the transmitters when used on the other scenarios (only 61.5% and 54.1% are obtained). It is likely because when the network is learned on Plain conditions the static channels contribute to the learning and the network is not able to focus on the radio properties themselves.

As expected however, the networks trained on more complex scenarios, is more robust and is efficient on less complex scenarios.

Typically, the network trained on the Varying dataset performs as well on the Plain and the Varying dataset. And the network trained on the Robot dataset performs equivalently on the three test datasets.

On the right Figure, the same behaviour are observed with MultiRx scenarios. But in addition, with the Robot, an accuracy loss is observed. When the signals are perturbed simultaneously by the channel gains, the robot and the receiver position, the learning conditions are the hardest ones. The fact that it is still possible to learn on these signals with an accuracy of 79% is quite encouraging.

C. Channel variations

All the previous results were obtained in FIT/CorteXlab under fixed conditions. As the human access to the shielded room is controlled, the environment was guaranteed unchanged during the experimentation.

Now, to evaluate the NN sensitivity to the environment, we introduce new test datasets. The environment in the shielded room is modified by adding a metallic chair in the room thus creating additional propagation paths. The formerly trained networks are then evaluated on these new datasets.

Fig. 11 presents the corresponding results where the accuracy obtained from the datasets before the environment perturbation are given in blue ( ) and the accuracy obtained on the datasets obtained after the perturbation are given in red ( ) for respectively MonoRx and MultiRx conditions.

Clearly, these results show the sensitivity of the network to the environment, especially when it learned on the Plain scenario. This kind of loss in accuracy has been already reported by other authors in [32], where they introduced artificial impairments to cope with it.

In our work, especially in MonoRx, we see how learning in more complete conditions (i.e. Robot scenario) allows to reduce significantly the accuracy loss. This is fully true in the MonoRx scenario, where the network learned with the Robot scenario is almost not sensitive to the environment perturbation. However, in MultiRx, the loss is reduced but remains present. Note that the 89.2% accuracy with the Robot-Robot test was obtained when the same dataset was split and used to learn and test, while the former section (see Fig. 10), the Robot-Robot test was using two different dataset but without deliberate environment perturbation. Unfortunately, the gathering of a Robot dataset currently requires human intervention to setup the robot and and this can lead to small involuntary channel perturbations. This means that the robot results from the previous section cannot be directly compared to the training results of Fig. 11 but this the channel change results. Therefore, in the result herein obtained we may assume that some

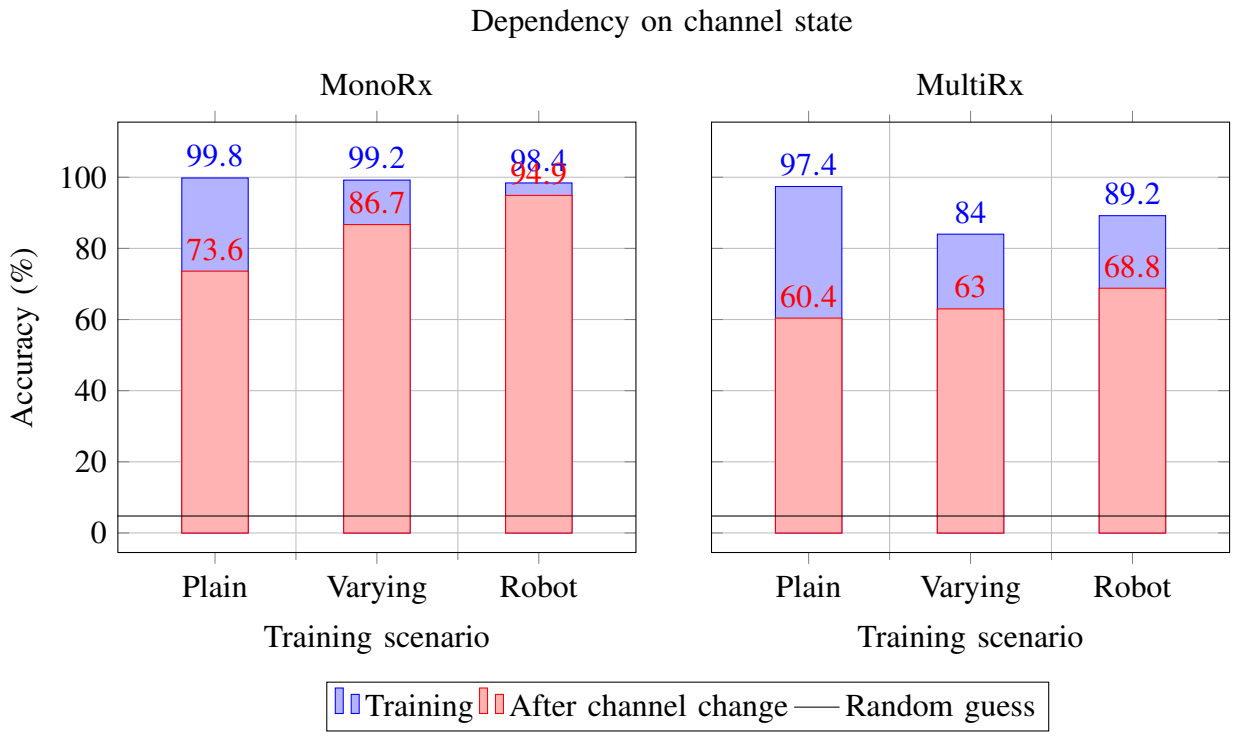

Fig. 11. Accuracy of networks trained on one scenario with static payloads and tested, either on data from the training dataset or on a dataset with the same scenario but with a modified environment.

information related to the channel is still used by the network to learn. However, we believe that the remaining accuracy clearly indicates that a part of the learning is performed on the RF signature and not on the channel conditions.

This is an encouraging result as it motivates the development of identification techniques based on RF signatures, and the dataset built in this work is unique in the sens that channel conditions are carefully controlled.

V. Additional notes on synchronisation

The FIT/CorteXlab platform contains a synchronisation system based on four octo-clocks [41] in a tree structure (not used in the previous section). The activation of said system produces a important reduction in generalisation performance, for every studied scenario, from 20pp to 60pp in accuracy. The goal of this section is to study the source of this performance drop. This synchronisation has two main components: at sample level, where packet start can be detected by the software an integer number of samples early or late, and sub-sample level where the hardware sampling time of the receiver doesn't match the one of the transmitting node.

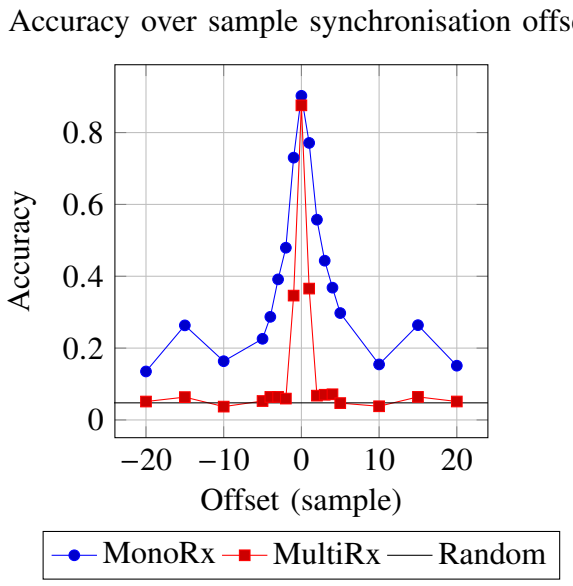

Fig. 12. Accuracy attained when the frame detector produces a timing offset

A. *Sample level synchronisation*

For both MonoRx and MultiRx, a standard Static Varying Scenario is gathered with no timing offset, and a network is trained on it. Then, a series of small datasets of the same scenario with various offsets is created. This is done by explicitly telling the payload extractor to record early or late.

Fig. 12 presents the effects of these synchronisation errors on a NN not trained to cope with them. Both settings clearly show a sharp decrease in accuracy. For MultiRx scenario, the accuracy level drops out even for an offset of 1 sample, while for MonoRx, the accuracy remains significant up to an offset of 3 samples. This effect may impact the performance in low SNR conditions, where sample synchronisation errors are more frequent.

B. *Sub-sample synchronisation*

Sub-sample synchronisation is a hardware effect so it needs to be simulated to allow study. A MonoRx Static Varying Scenario is gathered, with a payload oversampled by a factor of 16. To stay inside the signal processing capabilities of the hardware used, this is done by reducing the baud rate instead of increasing sample rate. Before feeding it to the neural network, this signal is separated into 16 undersampled time series with one sample taken every 16 and a starting offset between 0 and 15. Then, a network is trained on the undersampled series with an offset equal to 8 and tested on all the other offsets for four different datasets of the same scenario.

The decrease in classification accuracy with sampling time offset seen in Fig. 13 shows that a synchronised "naive" system is indeed unable to cope with sampling time variations. The small accuracy difference on offset 8 between the training dataset (2) and the others can be attributed to an additional, non artificial, offset smaller than the sampling time.

The output of this section is that when all nodes and Rx are synchronised, the NN cannot generalise. The reason follows: when the nodes are synchronised to the Rx, a timing offset exists that is constant over a dataset. But

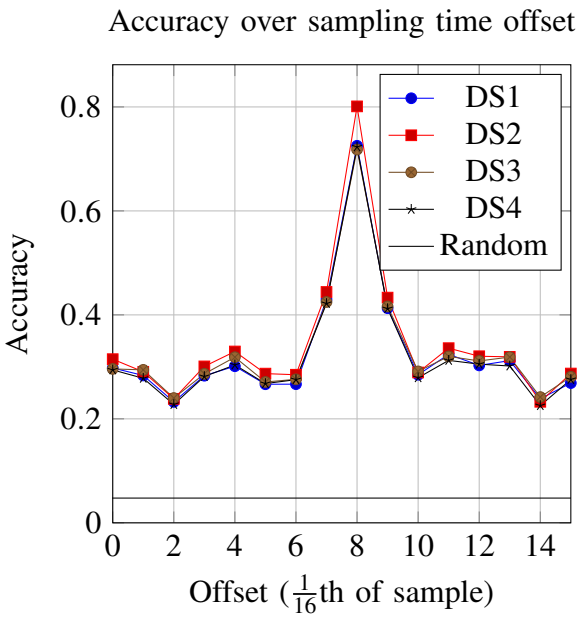

Fig. 13. Accuracy of a network trained on dataset 2 with an offset of eight and tested over the four datasets and 16 possible offsets

TABLE IV
ACCURACY OF NETWORKS TRAINED ON ONE SYNCHRONISATION POSSIBILITY AND TESTED ON THE OTHERS

Synced radio	Tx	Both	Rx	None
Tx	81.2%	34.5%	19.2%	38.3%
Both	54.4%	45.6%	38.2%	38.2%
Rx	21.1%	28.5%	80.2%	22.3%
None	41.3%	43.8%	34.6%	81.1%

this static timing offset changes by a random amount when a new acquisition starts. The NN learned on a dataset with a fixed sampling offset and is not able to adapt to a different one. On the other hand, as long as there is no synchronisation between nodes and receiver, the network can generalise since there, the offsets change even in the training data, and the network can learn to cope with it.

VI. Conclusion

A. Conclusion

This work explored a range of signal parameters that can impact the ability of a neural network to identify transmitters based on their physical layer characteristics and to generalise to more realistic environment, such as changing channel characteristics. The resulting guidelines to get a good transmitter identification is as follows: The part of a transmitted packet used for identification should be as much deterministic as possible, the physical preamble being a prime candidate. The channel encountered in the training dataset should have as much randomness as possible to force the network to learn to cope with a more realistic situation, and finally, it is not necessary to perfectly synchronise transmitters and receiver since this will only result in a poor generalisation performance.

The reader is invited to reproduce and extend these result by reusing the generated datasets and creating new ones in the FIT/CorteXlab or elsewhere using the data collection process [40] developed for this work and available online alongside the datasets.

B. Future works

To envision an industry grade implementation, several additional aspects still need to be studied:

QPSK modulation was chosen in this work as a realistic modulation example for the envisioned IoT usage of this technology. Perhaps other modulation schemes allow a more effective classification by exposing more dependable patterns or forcing amplifiers into less linear parts of their operating range such as a high peak-to-average power ratio (PAPR) OFDM setup. A study of what makes the best modulation scheme for this task could help improve the community's understanding on the matter.

Similarly, the bit sequence for the Static payload was chosen as a realistic example of an IoT type preamble bit sequence. Custom preambles could be crafted to simultaneously optimise classification accuracy and generalisation and frame detection and synchronisation performance.

The influence of the example size in term of number of samples has seen some attention in [16], but, apart from that, all the works in the literature use a different sample amount in examples. A more detailed study of this factor could help in comparing the existing works and provide guidelines and trade-offs for future implementations.

The current system setup is relevant in cases where all the users are known in advance, but it cannot handle the arrival of a new node, or the use of samples from new receivers. Architectures able either to detect intruders or erroneous users, or to add new users to the pool of known ones would be more versatile and allow for more dynamic use cases. A system able to agnostically use samples from any new receiver would greatly improve on the scalability and ease of field implementation.

The present paper purposely avoids reliance on channel effects to allow identification regardless of position, movements and environment changes. But these can be of use in the related task of location verification [42], and could be coupled with authentication in slow varying environments or very short-term identity verification.

Finally, one of the main promises of this technology is an increase in security from a reduced ability for attackers to spoof the identity of legitimate users. However, neural networks have been shown to easily suffer from adversarial examples, both in general [43], and on the specific topic of wireless communications [44]. So, to allow real implementation, any candidate architecture needs to show some adversarial robustness both in simulation and in the field.

REFERENCES

[1] D. Rupprecht, K. Jansen, and C. Pöpper, "Putting LTE security functions to the test: A framework to evaluate

implementation correctness," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. Austin, TX: USENIX Association, 2016.

[2] F. Fund. Run a man-in-the-middle attack on a wifi hotspot. [Online]. Available: https://witestlab.poly.edu/blog/conduct-a-simple-man-in-the-middle-attack-on-a-wifi-hotspot/

[3] M. Mitev, A. Chorti, M. Reed, and L. Musavian, "Authenticated secret key generation in delay-constrained wireless systems," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–29, 2020.

[4] K. Ellis and N. Serinken, "Characteristics of radio transmitter fingerprints," *Radio Science*, vol. 36, no. 4, pp. 585–597, 2001.

[5] Y. Chen, H. Wen, J. Wu, H. Song, A. Xu, Y. Jiang, T. Zhang, and Z. Wang, "Clustering based physical-layer authentication in edge computing systems with asymmetric resources," vol. 19, no. 8, p. 1926. [Online]. Available: https://www.mdpi.com/1424-8220/19/8/1926

[6] L. Senigagliesi, M. Baldi, and E. Gambi, "Statistical and machine learning-based decision techniques for physical layer authentication." [Online]. Available: http://arxiv.org/abs/1909.07969

[7] X. Wang, P. Hao, and L. Hanzo, "Physical-layer authentication for wireless security enhancement: Current challenges and future developments," *IEEE Communications Magazine*, vol. 54, no. 6, pp. 152–158, jun 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7498103/

[8] L. Xiao, L. J. Greenstein, N. B. Mandayam, and W. Trappe, "Using the physical layer for wireless authentication in time-variant channels," *IEEE Transactions on Wireless Communications*, vol. 7, no. 7, pp. 2571–2579, 2008.

[9] L. Xiao, L. Greenstein, N. Mandayam, and W. Trappe, "Fingerprints in the ether: Using the physical layer for wireless authentication," in *2007 IEEE International Conference on Communications*. IEEE, 2007, pp. 4646–4651.

[10] A. Weinand, M. Karrenbauer, R. Sattiraju, and H. Schotten, "Application of machine learning for channel based message authentication in mission critical machine type communication," in *European Wireless 2017; 23th European Wireless Conference*. VDE, 2017, pp. 1–5.

[11] L. J. Wong, W. C. Headley, and A. J. Michaels, "Emitter identification using cnn iq imbalance estimators," *arXiv preprint arXiv:1808.02369*, 2018.

[12] G. Li, J. Yu, Y. Xing, and A. Hu, "Location-invariant physical layer identification approach for WiFi devices," vol. 7, pp. 106 974–106 986.

[13] O. Ureten and N. Serinken, "Wireless security through RF fingerprinting," vol. 32, no. 1, pp. 27–33.

[14] X. Wang, J. Duan, C. Wang, G. Cui, and W. Wang, "A radio frequency fingerprinting identification method based on energy entropy and color moments of the bispectrum," in *2017 9th International Conference on Advanced Infocomm Technology (ICAIT)*, pp. 150–154, ISSN: null.

[15] N. Hu and Y.-D. Yao, "Identification of legacy radios in a cognitive radio network using a radio frequency fingerprinting based method," in *2012 IEEE International Conference on Communications (ICC)*, pp. 1597–1602, ISSN: 1550-3607.

[16] S. S. Hanna and D. Cabric, "Deep learning based transmitter identification using power amplifier nonlinearity," in *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2019, pp. 674–680.

[17] Y. Pan, S. Yang, H. Peng, T. Li, and W. Wang, "Specific emitter identification based on deep residual networks," vol. 7, pp. 54 425–54 434.

[18] B. Chatterjee, D. Das, and S. Sen, "Rf-puf: Iot security enhancement through authentication of wireless nodes using in-situ machine learning," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2018, pp. 205–208.

[19] A. Aghnaiya, A. M. Ali, and A. Kara, "Variational mode decomposition-based radio frequency fingerprinting of bluetooth devices," vol. 7, pp. 144 054–144 058.

[20] A. Candore, O. Kocabas, and F. Koushanfar, "Robust stable radiometric fingerprinting for wireless devices," in *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 43–49, ISSN: null.

[21] S. Chen, F. Xie, Y. Chen, H. Song, and H. Wen, "Identification of wireless transceiver devices using radio frequency (RF) fingerprinting based on STFT analysis to enhance authentication security," in *2017 IEEE 5th International Symposium on Electromagnetic Compatibility (EMC-Beijing)*, pp. 1–5, ISSN: null.

[22] G. Baldini, G. Steri, R. Giuliani, and C. Gentile, "Imaging time series for internet of things radio frequency fingerprinting," in *2017 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–6, ISSN: 2153-0742.

[23] G. Baldini, R. Giuliani, and F. Dimc, "Physical layer authentication of internet of things wireless devices using convolutional neural networks and recurrence plots," vol. 2, no. 2, p. e81. [Online]. Available: http://doi.wiley.com/10.1002/itl2.81

[24] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, 2018.

[25] H. Jafari, O. Omotere, D. Adesina, H.-H. Wu, and L. Qian, "IoT devices fingerprinting using deep learning," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, pp. 1–9, ISSN: 2155-7578.

[26] K. Youssef, L. Bouchard, K. Haigh, J. Silovsky, B. Thapa, and C. Vander Valk, "Machine learning approach to rf transmitter identification," *IEEE Journal of Radio Frequency Identification*, vol. 2, no. 4, pp. 197–205, 2018.

[27] I. O. Kennedy, P. Scanlon, F. J. Mullany, M. M. Buddhikot, K. E. Nolan, and T. W. Rondeau, "Radio transmitter fingerprinting: A steady state frequency domain approach," in *2008 IEEE 68th Vehicular Technology Conference*, pp. 1–5, ISSN: 1090-3038.

[28] K. Merchant, S. Revay, G. Stantchev, and B. Nousain, "Deep learning for rf device fingerprinting in cognitive communication networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 160–167, 2018.

[29] J. Yu, A. Hu, F. Zhou, Y. Xing, Y. Yu, G. Li, and L. Peng, "Radio frequency fingerprint identification based on denoising autoencoders," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2019, pp. 1–6.

[30] J. Yu, A. Hu, G. Li, and L. Peng, "A robust RF fingerprinting approach using multisampling convolutional neural network," vol. 6, no. 4, pp. 6786–6799.

[31] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu, and Y. Yan, "Design of a hybrid RF fingerprint extraction and device classification scheme," vol. 6, no. 1, pp. 349–360.

[32] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "Oracle: Optimized radio classification through convolutional neural networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 370–378.

[33] K. Sankhe, M. Belgiovine, F. Zhou, L. Angioloni, F. Restuccia, S. D'Oro, T. Melodia, S. Ioannidis, and K. Chowdhury, "No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 165–178, 2019.

[34] F. Restuccia, S. D'Oro, A. Al-Shawabka, M. Belgiovine, L. Angioloni, S. Ioannidis, K. Chowdhury, and T. Melodia, "DeepRadioID: Real-time channel-resilient optimization of deep learning-based radio fingerprinting algorithms," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '19. Association for Computing Machinery, pp. 51–60. [Online]. Available: https://doi.org/10.1145/3323679.3326503

[35] T. Oyedare and J.-M. J. Park, "Estimating the required training dataset size for transmitter classification using deep learning," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, pp. 1–10.

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[37] A. Massouri, L. Cardoso, B. Guillon, F. Hutu, G. Villemaud, T. Risset, and J.-M. Gorce, "Cortexlab: An open FPGA-based facility for testing SDR & cognitive radio networks in a reproducible environment," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*. IEEE, 2014, pp. 103–104.

[38] L. Sampaio Cardoso, O. Oubejja, G. Villemaud, T. Risset, and J. M. Gorce, “Reliable and Reproducible Radio Experiments in FIT/CorteXlab SDR testbed: Initial Findings,” in *Crowncom*, Lisbon, Portugal, Sep. 2017. [Online]. Available: https://hal.inria.fr/hal-01598491

[39] A. Mouaffo, L. Cardoso, H. Boeglen, G. Villemaud, and R. Vauzelle, “Radio link characterization of the CorteXlab testbed with a large number of software defined radio nodes,” in *Antennas and Propagation (EuCAP), 2015 9th European Conference on* , Lisbon, Portugal, Apr. 2015. [Online]. Available: https://hal.inria.fr/hal-01245107

[40] C. Morin. Datasets and data gathering code. [Online]. Available: https://wiki.cortexlab.fr/doku.php?id=tx-id

[41] E. Research. Octoclock product overview. [Online]. Available: https://www.ettus.com/wp-content/uploads/2019/01/Octoclock_Spec_Sheet.pdf

[42] S. Tomasin, A. Brighente, F. Formaggio, and G. Ruvoletto, *Physical-Layer Location Verification by Machine Learning*. John Wiley & Sons, Ltd, 2019, ch. 20, pp. 425–438.

[43] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks.” [Online]. Available: http://arxiv.org/abs/1312.6199

[44] B. Flowers, R. M. Buehrer, and W. C. Headley, “Evaluating adversarial evasion attacks in the context of wireless communications,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1102–1113, 2019.